Build CNN Model for Classification of Flowers

1. Download the Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd /content/drive/MyDrive
```

```
/content/drive/MyDrive
```

```
!unzip Flowers-Dataset.zip
```

```
Archive:  Flowers-Dataset.zip
replace flowers/daisy/100080576_f52e8ee070_n.jpg? [y]es, [n]o, [A]ll,
[N]one, [r]ename: N
```

```
pwd
```

```
{"type":"string"}
```

2. Image Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizon
tal_flip=True,vertical_flip=False)
```

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
pwd
```

```
{"type":"string"}
```

```
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/
flowers",target_size=(64,64),class_mode='categorical',batch_size=24)
```

```
Found 4317 images belonging to 5 classes.
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

3. Create Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D,Flatten,Dense
```

```
model=Sequential()
```

4. Add Layers(Convolution, MaxPooling, Flatten)

```
model.add(Convolution2D(32,
(3,3),input_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

```
32*(3*3*3+1)
```

```
896
```

Dense - (Hidden Layers)

```
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

Output Layers

```
model.add(Dense(5,activation='softmax'))
```

5. Compile the model

```
model.compile(loss='categorical_crossentropy',metrics=['accuracy'],opt
imizer='adam')

len(x_train)
```

```
180
```

```
4317/24
```

```
179.875
```

6. Fit the Model

```
model.fit(x_train, epochs = 5, validation_data=x_test,
steps_per_epoch=len(x_train), validation_steps=len(x_test))
```

```
Epoch 1/5
180/180 [==============================] - 1504s 8s/step - loss:
1.2941 - accuracy: 0.4498 - val_loss: 1.0301 - val_accuracy: 0.5844
Epoch 2/5
180/180 [==============================] - 64s 353ms/step - loss:
1.0419 - accuracy: 0.5877 - val_loss: 1.1322 - val_accuracy: 0.5650
Epoch 3/5
180/180 [==============================] - 63s 352ms/step - loss:
0.9456 - accuracy: 0.6375 - val_loss: 0.8333 - val_accuracy: 0.6861
Epoch 4/5
180/180 [==============================] - 65s 364ms/step - loss:
0.8759 - accuracy: 0.6579 - val_loss: 0.8886 - val_accuracy: 0.6697
Epoch 5/5
180/180 [==============================] - 64s 355ms/step - loss:
0.8099 - accuracy: 0.6894 - val_loss: 0.9797 - val_accuracy: 0.6396

<keras.callbacks.History at 0x7f6d37bce210>
```

7. Save the Model

```
model.save('flowers.h5')
```

```
ls flowers/
```

```
daisy/  dandelion/  rose/  sunflower/  tulip/
```

8. Test the Model

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

model=load_model('flowers.h5')

img=image.load_img(r"/content/drive/MyDrive/flowers/daisy/
100080576_f52e8ee070_n.jpg")

img
```

```
img=image.load_img(r"/content/drive/MyDrive/flowers/daisy/
100080576_f52e8ee070_n.jpg", target_size=(64,64))
```

```
img
```

```
x=image.img_to_array(img)
```

```
x
```

```
array([[[141., 141., 139.],
        [149., 149., 149.],
        [152., 152., 154.],
        ...,
        [162., 161., 166.],
        [154., 154., 152.],
        [153., 153., 153.]],

       [[136., 135., 131.],
        [146., 145., 143.],
        [169., 168., 174.],
        ...,
        [159., 158., 163.],
        [155., 155., 153.],
        [149., 149., 149.]],
```

```
        [[125., 125., 117.],
         [138., 140., 137.],
         [152., 152., 152.],
         ...,
         [156., 156., 156.],
         [157., 157., 155.],
         [143., 142., 140.]],

        ...,

        [[ 41.,  44.,  23.],
         [ 43.,  46.,  25.],
         [ 49.,  51.,  37.],
         ...,
         [128., 124., 121.],
         [125., 121., 118.],
         [125., 122., 117.]],

        [[ 43.,  46.,  25.],
         [ 43.,  46.,  25.],
         [ 54.,  55.,  37.],
         ...,
         [130., 126., 125.],
         [129., 125., 124.],
         [127., 123., 122.]],

        [[ 44.,  47.,  26.],
         [ 45.,  48.,  27.],
         [ 53.,  55.,  34.],
         ...,
         [137., 133., 132.],
         [133., 129., 128.],
         [130., 126., 125.]]], dtype=float32)
```

```
x=np.expand_dims(x,axis=0)
```

```
x
```

```
array([[[[141., 141., 139.],
         [149., 149., 149.],
         [152., 152., 154.],
         ...,
         [162., 161., 166.],
         [154., 154., 152.],
         [153., 153., 153.]],

        [[136., 135., 131.],
         [146., 145., 143.],
         [169., 168., 174.],
```

```
          ...,
          [159., 158., 163.],
          [155., 155., 153.],
          [149., 149., 149.]],

         [[125., 125., 117.],
          [138., 140., 137.],
          [152., 152., 152.],
          ...,
          [156., 156., 156.],
          [157., 157., 155.],
          [143., 142., 140.]],

         ...,

         [[ 41.,  44.,  23.],
          [ 43.,  46.,  25.],
          [ 49.,  51.,  37.],
          ...,
          [128., 124., 121.],
          [125., 121., 118.],
          [125., 122., 117.]],

         [[ 43.,  46.,  25.],
          [ 43.,  46.,  25.],
          [ 54.,  55.,  37.],
          ...,
          [130., 126., 125.],
          [129., 125., 124.],
          [127., 123., 122.]],

         [[ 44.,  47.,  26.],
          [ 45.,  48.,  27.],
          [ 53.,  55.,  34.],
          ...,
          [137., 133., 132.],
          [133., 129., 128.],
          [130., 126., 125.]]]], dtype=float32)
```

```
y=np.argmax(model.predict(x),axis=0)
```

```
y
```

```
array([0, 0, 0, 0, 0])
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
index=['daisy','dandelion','rose','sunflower']
```

```
index[y[0]]
```

```
{"type":"string"}
```

```
img=image.load_img(r"/content/drive/MyDrive/flowers/dandelion/
10200780773_c6051a7d71_n.jpg", target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['daisy','dandelion','rose','sunflower']
index[y[0]]
```

```
{"type":"string"}
```

```
img
```

```
img=image.load_img(r"/content/drive/MyDrive/flowers/sunflower/
10386503264_e05387e1f7_m.jpg", target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=0)
index=['sunflower','daisy','dandelion','rose']
index[y[0]]
```

```
{"type":"string"}
```

```
img
```